The user interface circuitry can also include circuitry for providing audible signals to the user through a tone generator.

Page 17, lines 8-13, delete current paragraph and insert therefor:

A2

Although the invention could be implemented in various ways, the invention has been successfully implemented by programming computing devices to employ knowledge brokers and feature constraints as described by Andreoli et al., above. A demonstration of a prototype can be viewed at the web site page "www.xrce.xerox.com/research/ct/projects/cbkb/home.html." This section reviews relevant aspects of knowledge brokers and feature constraints.

Page 25, lines 5-6, delete current paragraph and insert therefor:

A3

These axioms form a theory *T*, for example, and are formally written, for example, as the following three sets of axioms.

Page 25, between lines 6 and 7 insert a new paragraph as follows:

A4

Specific axioms for features and sorts:

Let $\tau$, $\tau^1$ denote any sorts, and $f$ denote any feature.

$$\forall x, y, z \quad x \xrightarrow{f} y \wedge x \xrightarrow{f} z \supset y = z$$

$$\forall x \quad \neg(x : \tau \wedge x : \tau') \text{ if } \tau \neq \tau'$$

$$\forall x, y \quad x : \tau \wedge y : \tau \supset x = y \text{ if } \tau \text{ is a value sort}$$

$$\forall x, y \quad \neg(x : \tau \wedge x \xrightarrow{f} y) \quad \text{if} \quad \tau \text{ is a value sort}$$

Congruence axioms for equality:

Let $p$ denote any built-in predicate. The traditional congruence axioms are, for example:

$$\forall x \qquad x = x$$

$$\forall x, y \qquad x = y \supset y = x$$

$\forall x, y, z \qquad x = y \wedge y = z \supset x = z$

$\forall x, \ y \qquad z : \tau \wedge x = y \supset y : \tau$

$\forall x, y, z \qquad x \xrightarrow{f} y \wedge x = z \supset z \xrightarrow{f} y$

$\forall x, y, z \qquad x \xrightarrow{f} y \wedge y = z \supset x \xrightarrow{f} z$

$\forall \bar{x}, y \qquad p(\bar{x}) \wedge x_i = y \supset p(\bar{y})$

where i is some index in the list of variable $(\bar{x})$ and $(\bar{y})$ is identical to $(\bar{x})$ except that $y_i = y$.

Built-in predicate axioms:

They must not mention sorts and features. For example, disequality can be axiomatized by

$\forall x,y \quad x \neq y \vee x = y$

$\forall x \qquad \neg(x \neq x)$

Precedence constraints are axiomatized by

$\forall x \ \neg(x < x)$

$\forall x, y, z \qquad x < y \wedge y < z \supset x < z$

The built-in predicates $>, \leq, \geq$ can then be defined from $<$ and equality.

Page 25, lines 13-15, delete current paragraph and insert therefor:

Constraint satisfaction over BFCs is defined by a set of conditional rewrite rules over BFCs. For example, where a BFC is represented as a pair $(B \mid \Gamma)$ where $B$ is a built-in constraint and $\Gamma$ an unordered list of sort and feature constraints (read conjunctively), the following rules, for example, correspond to simplifications of the BFCs. In the rules set forth below, $\perp$ denotes the contradiction. The set of conditional rewrite rules include:

$(B \mid x \xrightarrow{f} y, = 1 \xrightarrow{f} t, \Gamma) \rightarrow (B \wedge y = t \mid x \xrightarrow{f} y, \Gamma)$ if

$\vdash \tau\ B \supset x = z$

$(B \mid x: \tau, y: \tau, \Gamma) \rightarrow (B \mid x: \tau, \Gamma)$ if $\vdash \tau\ B \supset x = y$ and $\tau$ is not a value sort.

$(B \mid x: \tau, y: \tau, \Gamma) \rightarrow (B \wedge x = y \mid x: \tau, \Gamma)$ if $\tau$ is a value sort.

The following rules, for example, correspond to the detection of inconsistencies:

$(B \mid \Gamma) \rightarrow \perp$ if $\vdash \tau\ \neg B$

$(B \mid x: \tau, y, : \tau', \Gamma) \rightarrow \perp$ if $\vdash \tau\ B \supset x = y$ and $\tau \neq \tau'$

$(B \mid x: \tau, y \xrightarrow{f} z, \Gamma) \rightarrow \perp$ if $\vdash \tau\ B \supset x = y$ and $\tau$ is a value sort.

The following property justifies the algorithm

$(B \mid \Gamma) \rightarrow \perp$ if and only if $\vdash \tau\ \forall \neg (B \bigwedge_{c \in \Gamma} c)$.

Page 25, between lines 15 and 16, insert a new paragraph as follows:

These rules have the following properties:

Page 26, lines 14-22, delete current paragraph and insert therefor:

The algorithm for constraint satisfaction over SFCs can informally be described with the following example. For example, an SFC is represented as an *unordered list* of BFCs prefixed with a sign (+ or -); by definition, one and only one component is positive. Let $S$ be an SFC. The SFC-normal form of $S$ is written $S$ and is obtained by the following algorithm:

Let $c_o$ be the BFC-normal form of the positive component of $S$.

If $c_o = \perp$ Then

Return $\perp$

Else

$C_o$ is f the form $(B_o \mid \Gamma_o)$

Let $\{(Bi \mid \Gamma_i)\}i = 1,\ldots,n$ be the list of negative components of $S$.

For each i = 1, ..., $n$

Let $c_i$ be the BFC normal form of $(B_o \wedge B_i \mid \Gamma)$.

If there exists $\underline{i} \in 1, \ldots, n$ such that $c_i = (B \mid \Gamma)$ and $\tau\ B$ and $\Gamma$ is empty Then

Return $\perp$

Else

Let $I = \{i \in 1, \ldots, n$ such that $c_i \neq \perp\}$

Return $\{+c_{o1}\ \{-c_i\}i \in 1\}$.

The following property justifies the algorithm:

$$[+(B_o \mid \Gamma_o), \{-(B_i \mid \Gamma_i)\}\ \frac{n}{i=1} = \perp \text{ if and only if } \tau\ \forall\neg[(B_o \wedge \bigwedge_{c \in \Gamma_o} c) \wedge \bigwedge_{i=1}^{n} \neg(B_i \wedge \bigwedge_{c \in \Gamma_i}^{n} c)].$$

Therefore, for example, given an SFC, its positive component is first normalized by the algorithm for BCFs. If the result is a contradiction, the whole SFC is unsatisfiable. Otherwise, the normalized positive component is inserted in each of the negative components, which are then normalized by the algorithm for BFCs. If a resulting negative component has a contradictory normal form, it is eliminated, and if it has a tautological normal form the whole SFC is unsatisfiable. The normal form for SFCs thus obtained has the following property:

Page 33, lines 18-22, delete current paragraph and insert therefor:

A document constraint descriptor could be transferred between two computing devices in the manner described in copending, coassigned U.S. Patent Application No.